SOFTWARE ENGINEERING AND SIMULATION CREDIBILITY

Ronald L. Ketcham and Paul R. Muessig
Naval Air Warfare Center, Weapons Division (Code 418000D)
1 Administration Circle
China Lake, CA 93555-6100
Ketchamrl@navair.navy.mil, muessigpr@navair.navy.mil

KEY WORDS

Software Engineering, Software Quality Assurance, Configuration Management, Requirements Management

ABSTRACT

Most people think of "validation" as the hallmark of simulation credibility. But some simulations, by their very nature (e.g., mission level models, highly complex physics-based simulations, etc.) are notoriously difficult to validate. There are also situations in which the process of validation, even if feasible, cannot keep pace with the dynamic nature of simulation evolution, or where the cost of validation data is prohibitively high. Are there any other "hallmarks" of simulation credibility that can help us in cases like these? This paper demonstrates that significant insights into simulation credibility can be obtained via detailed examination of the software engineering processes and practices by which the simulation was developed and is maintained. The nature and value of good software engineering processes is often unknown, or under-appreciated, by the end-users of simulation results and therefore is often overlooked as a barometer of simulation credibility.

INTRODUCTION

To explore this topic we will use DoD mission-level models as a starting point. These models have unique characteristics when compared to other model categories on the DoD M&S model pyramid[1] which make them exemplars of simulations that have rapidly changing requirements and that are difficult to validate. Of all the model categories included, mission-level models have the widest range of applications, a fact that has resulted in these simulations having large and diverse user bases. The DoD has three mission-level models in wide use today: Suppressor, JIMM (Joint Interim Mission Model, formerly known as SWEG), and EADSIM. These models are used in a wide variety of applications by the acquisition and analysis communities. They are used to evaluate the potential effectiveness of new systems and platforms, different tactical employments, and even various command and control approaches. Recently, these models have also seen growth in the user base among the testing and training communities. This great diversity of users and applications means that these mission-level models are constantly called upon to meet new requirements at an ever-increasing rate. Suppressor and JIMM typically have over 100 new requirements specified over a two-year period. The dynamic nature of these models has emphasized the value of good model management and software development processes.

The second characteristic mentioned above is the difficulty of validating these models. First, they can never be validated in their entirety because the data will never exist. Individual functions within the simulation may be validated, but validation of their integration would require data from actual large-scale combat situations. Such data are very difficult to come by, and are rarely collected in actual combat given the obvious press of higher priorities. Also, there are human factors incorporated in mission-level models that pose serious validation problems. For example, we cannot measure human reaction times in the fog of war on a test range; the environment is simply too artificial. Moreover, the value of any structured validation data that might be collected is diminished in a test situation due to human safety considerations. This limits us to validation of mission model functional components. For example, we can get data to validate sensor detection ranges or examine missile kinematics. Finally, validation efforts have a limited shelf life for mission-level models. As these highly dynamic models change over time, validation results become obsolete. Due to these limitations, it was not surprising to us

[1] See, for example, the Air Force Standard Analysis Toolkit web site: http://www.oas.kirtland.af.mil/restrict/toolkit/index.html

Report Documentation Page

Report Date	Report Type	Dates Covered (from... to)
00AUG2001	N/A	-

Title and Subtitle Software Engineering and Simulation Credibility	**Contract Number**
	Grant Number
	Program Element Number
Author(s) Paul R. Muessig, Ronald L. Ketcham	**Project Number**
	Task Number
	Work Unit Number
Performing Organization Name(s) and Address(es) Naval Air Warfare Center, Weapons Division (Code 418000D)1 Administration Circle, China Lake, CA 93555	**Performing Organization Report Number**
Sponsoring/Monitoring Agency Name(s) and Address(es)	**Sponsor/Monitor's Acronym(s)**
	Sponsor/Monitor's Report Number(s)

Distribution/Availability Statement
Approved for public release, distribution unlimited

Supplementary Notes

Abstract
see report

Subject Terms

Report Classification unclassified	**Classification of this page** unclassified
Classification of Abstract unclassified	**Limitation of Abstract** SAR

Number of Pages
7

to find in a recent review that both Suppressor and JIMM lack any current validation results in support of their credibility, despite their wide employment.[2]

Are there any other indicators of simulation credibility that can be employed when little, or no, validation data exist, or when the requirements generation environment is very dynamic? We propose that an evaluation of the software engineering processes employed to develop and maintain the simulation can provide insight into the credibility of the simulation. We will use the SEI Capability Maturity Model to examine this issue.

CAPABILITY MATURITY MODEL

In the early 1990's, the Software Engineering Institute (SEI) at Carnegie-Mellon University developed the Capability Maturity Model (CMM). The primary purpose of the tool was to provide the Department of Defense (DoD) assistance in assessing the ability of software development organizations to deliver quality software products within cost and schedule constraints. The CMM is intended to provide the software development customer a means to evaluate the maturity of candidate software developers and their ability to perform as specified (Paulk ***et al***. 1995). The CMM provides a framework for identifying the maturity of the organization from level 1 (the most immature) to level 5 (the most mature). The Level 1 organization has no documented software development processes and executes a software development project primarily by writing code and putting out fires. The level 5 organization has well defined and documented processes addressing key software development activities, as well as a trained and disciplined development team that implements those processes. The level 5 organization even has established processes for identifying and fixing weaknesses within its development processes. Table 1 depicts the five levels of maturity for a software development organization. Each level, except Level 1, has unique Key Process Areas (KPAs) which characterize the activity focus for that level. To move from one level to another, a software development organization must demonstrate competence in all of the KPAs both at and below its level of maturity.

While the more mature software development organization may well develop quality software on time and within budget, but will it also develop a more credible simulation? Figure 1 provides some data that support this hypothesis.[3] It shows that defects per KSLOC (thousand lines of source code) decreases monotonically as CMM maturity level increases. A Level 1 organization averages 9 defects per KSLOC, while a Level 5 organization averages only 0.1 defects per KSLOC. The data also show that it takes a Level 5 organization about 1/4th the time as a Level 1 organization to produce software that has only 1/9th the number of errors in it. Put another way, a Level 1 organization spends four times as much time producing nine times as many errors as a Level 1 organization. This difference can be expected to have a profound impact on simulation credibility. The chart also shows that over 3/4ths of the software organizations surveyed were at or below CMM Level 2.

Level	CHARACTERISTICS	Key Process Areas
5 Optimizing	Continuous process capability improvement	Process change management Technology change management Defect Prevention
4 Managed	Quantitative measurement of process and qualitative management of product	Software Quality Management Quantitative Process Management
3 Defined	Software Processes Defined and Institutionalized	Peer Reviews Inter-group Coordination Software Product Engineer Integrated Software Management Training Program Organization Process Definition Organization Process Focus
2 Repeatable	Management Controls in place; stable planning and product baselines	Software Configuration Management Software Quality Assurance Requirements Management Software Project Tracking and Oversight Software Project Planning Software Subcontract Management
1 Initial	Ad-hoc, Putting out Fires	None

Table I: Capability Maturity Model Levels and KPAs

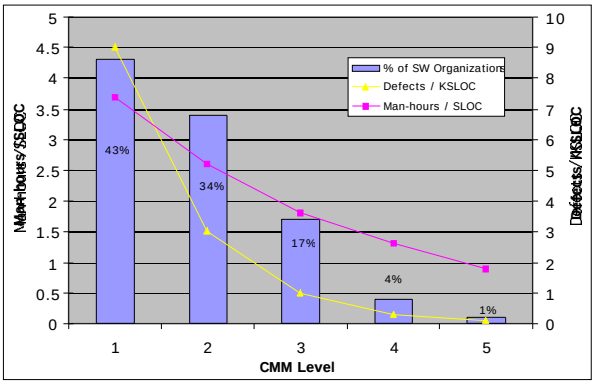

Figure 1: Software Coding Error Rates and Man-Hours per SLOC as a function of CMM Level

[2] See, for example, Accreditation Support Package, Level I for JIMM and Suppressor, available on the JASA web site (http:\\www.nawcwpns.navy.mil\~jasa).

[3] Briefing by RAdm. Joseph Dyer to the NAVAIR Senior Steering Group, 30 August 1999, "Software Implementation Team Status Report".

There are many ways that the software engineering processes used in the development and maintenance of a simulation can contribute to (or detract from) simulation credibility. One way is reflected in the ability of the processes used to capture and document the expertise of recognized Subject Matter Experts (SMEs) in each area of required simulation functionality, and to translate this expertise into documented simulation requirements. The process must be robust enough to accurately translate those requirements into simulation design, implement the design in software, and verify its correct implementation. Simulation credibility is directly related to the degree to which these tasks have been accomplished accurately. The CMM captures this sequence in the Key Process Areas of Requirements Management, Peer Reviews, Configuration Management, and Software Quality Assurance. The following section will detail these KPAs, the potential contribution each can make to simulation credibility, and the methods that can be used to assess if these factors are being adequately incorporated into simulation development and maintenance.

Requirements Management

Requirements definition is one of the most critical aspects of software development, and is a good indicator of the potential for success of the development effort in terms of schedule, cost, and software quality. On the surface this may appear to be a simple exercise, but it is difficult to implement correctly and comprehensively. The primary obstacle is that users (who must define and "own" the simulation requirements) and developers (who must implement these requirements in software) often have two different worldviews and speak in two different languages. For example, the user of a mission-level model views the simulation in terms of specific platforms, systems, tactics, and engagements; the developer of the simulation sees objects, data structures, algorithms and events. In many cases both participants are ignorant of the other's area of expertise or the necessary contribution that the other makes to simulation credibility.

According to the CMM, the purpose of Requirements Management (RM) is to establish and maintain a common understanding of simulation requirements between the developer and the end-user of the simulation under development. Well-documented requirements serve the same function as the Rosetta stone, enabling user requirements to be translated into terms understood by the developer, and enabling the developer's design to be translated into terms understood by the user.

Requirement definition is an inherently tedious and laborious task to the typical SME. A proper requirements document defines simulation requirements in sufficient detail to enable accurate implementation and verification of the simulation design. The less the developer understands the system or phenomenon being simulated, and the level of fidelity at which it must be simulated, the more a good requirements document becomes essential. In many cases the user does not understand or appreciate the need for this level of definition, and may not be inclined to assemble the requisite SME expertise to assist with requirements definition. Even if the right SMEs are assembled, they may lack an understanding of the developer's need for detail, as much as the developer often lacks experience in the simulated subject matter.

Requirements must originate from appropriate SMEs selected or approved by the user. Simulation credibility is only as good as the SMEs who define the requirements. While it is sometimes delegated to the developer to draft the initial simulation requirements, the user must not abdicate responsibility for (or ownership of) them. The user must fight to identify and overcome any communication barriers between SMEs and the developer, ignore the tedium, roll up his sleeves and ensure that the job gets done correctly. The developer can and should assist in this process of free and open communication and collaboration. He must ensure that the user's SMEs have supplied sufficient detail for the design phase of development.

At some point in the requirements definition process the "real world" gets abstracted. Planes and ships become "platforms"; radars, infrared detectors, and human eyes become "sensors". The best abstractions lead to more elegant and maintainable code. However, the essence of the subject matter must not be lost in the process of abstraction. The SME can ensure this does not happen by being an active participant in the requirements definition process. SMEs should participate in this process throughout the design phase of simulation development. Once simulation requirements are documented, the developer should produce design documents detailing how requirements are to be implemented in software.

Real world abstraction also takes place at this point. For example, Integrated Air Defense Systems become "acyclic graph data structures". Design documents should be drafted in a manner that the user can fully understand and comprehensively review. It must be insured that critical elements of the requirements are not lost in the design phase.

Requirements documentation should not only define the specific requirements, but should also specify the verification methods and success criteria to be used for each requirement. All requirements should be specified in such a manner as to be testable. If they cannot be tested, then how will their correct implementation be verified? The SME should assume ownership of the test description and specification for successful implementation of the design into software.

One of the best ways to mitigate risk in the requirements phase is to develop requirements incrementally. This has sometimes been referred to as "rapid prototyping". This is an especially useful approach to the development of complex simulations such as mission-level models. Requirements can be broken down into separate model builds, and drafted in separate development cycles. This allows SMEs to refine and add to requirements as each build, or prototype, is delivered, and as experience with (and insight into) the simulation is gained. The breakdown of requirements also facilitates using multiple panels of SMEs, recognizing that experts may only have expertise in particular subsets of the entire scope of the simulation. The key pitfall with the approach described above is requirements creep after requirements have been frozen. Once requirements have been frozen, new requirements should wait until the next development cycle.

Below are some of questions that can help a user to assess how well simulation requirements are being managed. As described above, the credibility of the resultant simulation is, in part, determined by the adequacy of the answers to these questions. They all give some indication as to whether the expertise of the recognized SMEs was adequately captured and correctly implemented in the simulation:

- Did appropriate and qualified SMEs generate simulation requirements? What is the breadth and depth of the collective experience of the SME panel? Do they have relevant community recognition or acceptance?
- Does the developer have simulation experience in the SME's area of expertise? (This helps facilitate a common understanding of simulation requirements between developers and SMEs.)
- To what extent did the SMEs participate in the generation of simulation requirements? Did they generate the requirements document, or was the task delegated to the developers? Did they review and comment on design documents?
- Are the simulation requirements well documented, comprehensive, and in sufficient detail for the developer to translate them into simulation design and code?
- Are the requirements testable? If so, are the tests well defined, and are success criteria documented with each requirement?
- Were requirements developed incrementally? How have the builds been divided? Has this process been used to refine requirements as the SME gains experience with the simulation? Has the requirement definition process been tailored to enhance the participation of focused SMEs?
- Were requirements frozen during design, implementation and testing? If not, how was the traceability of requirements maintained during development?

Peer Review

The Peer Review KPA also plays a significant role in ensuring simulation credibility. The goal of peer reviews is to prevent and remove defects in the simulation as early and as efficiently as possible. Peer reviews involve a methodical examination of software work products by the producers' peers to identify errors and other potential problems. For our purposes, we choose to include SMEs in the definition of peers, along with other software developers. The work products to be reviewed include requirements documents, design documents, test reports, and even the source code. When peer reviews are minimal, or nonexistent, the user runs substantial risk of the end-product not meeting requirements, or even worse, containing undetected errors.

As stated earlier, simulation credibility is enhanced when SMEs play a review role throughout

the development cycle. By reviewing design documentation, SMEs can verify that the implementation methods chosen by the developer are true to their defined requirements. This same logic applies to other software development artifacts, such as test reports.

Even non-SMEs can play a significant role in enhancing simulation credibility through peer reviews. Other programmers can detect common implementation errors during detailed code walkthroughs, such as incorrectly indexedloops or common mathematical equation errors. While some of these may be detected during testing, this is not always the case. Peer reviews of this nature can also help developers enforce coding standards and improve implementation techniques to enhance the long-term maintainability of the simulation software. This may be critical to enhancing credibility in simulations that continue to grow and evolve rapidly over time.

The following are some key questions to ask regarding peer reviews that are aimed at improving simulation credibility:

- Does the development process call for planned periodic peer reviews? If so, what types of reviews are conducted? How often?
- What software work products are reviewed? Are review criteria and goals explicitly articulated?
- Are SMEs encouraged to participate in the peer review process? Are review materials, such as design documents written with their areas of expertise in mind?
- Do SMEs utilize opportunities for peer review? Are they effectively integrated into the review process?
- How are reviewers concerns captured and addressed, and what products result from reviews? Are these products structured to assist in the assessment of simulation credibility?

Software Configuration Management

The next KPA considered is Software Configuration Management (SCM). The general purpose of SCM is to establish and maintain the integrity of the software products throughout the simulation's life cycle. Within this general definition there are many specific functions handled by SCM. Since their stated purpose is to maintain the integrity of the simulation, all of these functions when executed properly enhance the credibility of the simulation. Therefore, the existence of a rigorous and robust SCM process is a positive indicator of simulation credibility.

The scope of SCM activities is too broad to cover here in its entirety. Therefore, three aspects of the SCM process have been selected for further discussion.

Requirements Tracking. Requirements for simulations such as mission-level models change constantly. A key function of the SCM process is requirements tracking (as opposed to Requirements Management, discussed above). Once requirements are documented, either initially in a requirements document or later through Requirements Change Forms of some type, they are entered into the SCM system. Here their status is tracked through implementation, testing, and closure. Requirements tracking enhances simulation credibility by increasing the probability that simulation requirements specified by SMEs will get implemented in software in a controlled and disciplined manner. Users will better understand the capabilities of each new release of the simulation to meeting their needs.

Configuration Control Board (CCB). The CCB, in one form or another, is usually the managing authority over the simulation development process. The CCB should be populated with the stakeholders of the simulation, i.e., with those who have a vested interest in the future of the simulation. The CCB has many roles and responsibilities. In practice, these roles and responsibilities are sometimes divided up among multiple governing boards. For example, a Configuration Review Board (CRB) may be established to rank change requests (new requirements) by technical merit and need. This ranking is then sent to the CCB to be finalized based on programmatic (budget and cost) concerns. In whatever form this body takes, the CCB is a key forum to capture the interests of simulation users and their SMEs. A well run CCB with active participation by the SMEs will help to ensure that simulation development is managed in such a way to enhance credibility.

SCM Database. The SCM database contains all proposed requirements and known errors in the simulation, along with the status of each. At any given time, this database will give potential users insight into the limitations and errors contained

within the simulation. This allows the user to assess the impact of these limitations to a specific application.

The following are some key questions to ask regarding SCM that are aimed at improving simulation credibility:

- Is there a documented SCM process? Are the nature, scope and depth of activities robust enough to capture and control problems with software, documentation and databases?
- Are simulation requirements well tracked and controlled?
- Are known errors documented in the SCM database?
- What is the Change Request or Trouble Report submittal process? Is this process open and easily accessed by all users and SMEs?
- Are users and SMEs able to review the SCM database at all times?
- Is there an active CCB (or its equivalent)? Does its membership include users and SMEs? Do SMEs participate actively in the CCB?
- How well does the CCB function? Is it an active body providing the developer with good insight into user requirements, or does it function as a "rubber stamp" for the model manager?

Software Quality Assurance

The final KPA to be examined is Software Quality Assurance (SQA). The role of SQA is to ensure that all-relevant software development standards and processes are applied correctly. If this function is treated as a matter of faith, simulation managers and users may remain blind to many problems that exist within the software development effort. The following are some of the significant functions of SQA that have an impact on simulation credibility (Humphrey 1989):

- SQA ensures that a requirements traceability matrix, or similar device is used to show that simulation capabilities cover the requirements specified by SMEs.
- SQA ensures that an implementation traceability matrix, or similar device is used to show that SME-specified requirements are implemented in the design.
- SQA reviews documentation to ensure that all documents satisfy established standards and are kept current.
- SQA reviews all development artifacts to ensure they are being adequately maintained.
- SQA periodically reviews SCM to ensure it is maintaining proper baseline control as well as full change records for requirements, design, code, test, and documentation.
- SQA reviews all test reports to ensure all testing is performed as appropriate and that the simulation satisfies established acceptance criteria.
- SQA monitors all peer reviews to ensure they are conducted as planned, that the results are documented, and that all follow-up items are executed.

SQA is recognized as a specific discipline that is best served by a trained and dedicated staff. Simulation credibility is directly related to how well SQA verifies proper execution of all processes, specifically those that ensure the expertise of the SMEs are fully realized within the simulation.

SUMMARY

This paper has described the connection between good software development processes and enhancing simulation credibility. We have demonstrated this relationship using a subset of the CMM framework. However, the contribution of the CMM to simulation credibility is not limited to these four KPAs. Other KPAs, such as Software Quality Management (from CMM Level 4) and Defect Prevention (from CMM Level 5) may be of use in this regard.

Also, while the CMM appears to be a good approach to categorize and evaluate software development processes, it is not the only framework that may have merit as an indicator of simulation credibility. For example, Object Oriented Design and Analysis (OOD&A) can enhance the maintainability and extensibility of a proposed simulation. In a simulation that will undergo a rapid and continuous evolvement of requirements, OOD&A may significantly promote credibility if applied well.

We have demonstrated that there exists a causal link between simulation credibility and the quality of software engineering processes and resources applied. These insights can be applied by all parties involved in simulation development - the developer, the end-user, and the simulation evaluator - to enhance simulation credibility.

Although this paper has focused on mission-level models, it should be clear that all simulations could benefit from the relationship between good software development processes and simulation credibility as demonstrated here. All simulations will have enhanced credibility by being developed and maintained with software engineering practices that promote the implementation of the expertise of qualified SMEs, while at the same time preventing, identifying, and removing defects.

REFERENCES

Humphrey, W.S.. 1989. ***Managing the Software Process***. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts.

Paulk, M.C., Weber, C.V., Curtis, B., Chrissis, M.B., ***et al***. 1995. The Capability Maturity Model: Guidelines for Improving the Software Process. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts.

ABOUT THE AUTHORS

Ronald L. Ketcham

Mr. Ketcham is currently an Accreditation Support Analyst with the Joint Accreditation Support Activity (JASA) at the Naval Air Warfare Center, Weapons Division, China Lake, CA. He is currently the Lead Accreditation Agent for the Joint Strike Fighter (JSF) Program. Mr. Ketcham has extensive experience in software engineering and VV&A. He gained much experience in the CMM as Project Manager of the Analyst's Work-Bench (AWB) in the early 1990's. Mr. Ketcham holds an A.B. in Economics from Transylvania University (in Lexington, Ky.) and an M.S. in Economics and Industrial Engineering from Iowa State University. He is currently working on an M.S. in Computer Science with California State University at Chico.

Paul R. Muessig

Dr. Muessig is currently Director of the Joint Accreditation Support Activity (JASA) at the Naval Air Warfare Center, Weapons Division, China Lake, CA. JASA's goal is to support weapons system acquisition and testing programs with credible integration of M&S into program objectives. Dr. Muessig has extensive experience in risk-based VV&A methodology development, planning and execution in support of numerous M&S applications in acquisition and testing. He has also worked as a defense analyst at the Center for Naval Analyses in Washington, D.C., contributing to survivability analyses for advanced technology aircraft, and leading model validation efforts for the Advanced Low Altitude Radar Model (ALARM). Dr. Muessig holds a B.S. in Chemistry from St. Joseph's University in Philadelphia, and a doctorate in Physical Chemistry from Brown University in Providence, Rhode Island.